NOTICE: When government or other drawings, specifications or other data are used for any purpose other than in connection with a definitely related government procurement operation, the U. S. Government thereby incurs no responsibility, nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.
NONLINEAR EXTRAPOLATION AND TWO-POINT BOUNDARY VALUE PROBLEMS

Richard Bellman, Harriet Kagiwada and Robert Kalaba

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA
This research is sponsored by the United States Air Force under Project RAND—contract No. AF 49(638)-700 monitored by the Directorate of Development Planning, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

DDC AVAILABILITY NOTICE
Qualified requesters may obtain copies of this report from the Defense Documentation Center (DDC).
Many problems in modern control engineering involve the numerical solution of nonlinear differential equations subject to two-point boundary conditions. The purpose of this Memorandum is to show how nonlinear extrapolation techniques can be used to convert a first-order successive approximation scheme into a second-order scheme for the solution of such problems. Results of a numerical experiment are presented.
SUMMARY

It is suggested that the convergence properties of the usual Picard successive approximation scheme may be improved through use of nonlinear extrapolation techniques. A numerical example is provided.
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREFACE</td>
<td>iii</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>v</td>
</tr>
<tr>
<td>Section</td>
<td></td>
</tr>
<tr>
<td>I.  INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II.  SKETCH OF METHOD</td>
<td>2</td>
</tr>
<tr>
<td>III. EXAMPLE</td>
<td>3</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>4</td>
</tr>
<tr>
<td>FORTRAN II PROGRAMS</td>
<td>5</td>
</tr>
</tbody>
</table>
I. INTRODUCTION

An interesting class of nonlinear two-point boundary value problems is described by the following equations

\[ u'' = f(u), \quad u(0) = u(l) = 0. \]  \hspace{1cm} (1)

In the scalar case, the equation can be solved by quadrature; in the vector case, we must use numerical techniques. Let us restrain our attention to the scalar case in order to present a new approach unhindered by analytic details. One way to attack these problems is to use Newton's method and quasilinearization.\(^{(1)}\) Such an approach requires knowledge of the partial derivative \( \partial f/\partial u \) for the determination of the \((n+1)^{st}\) approximation to the solution, given an \(n^{th}\) approximation, \(u_n(x)\). The recurrence relations are the linear equations

\[ \begin{align*}
    u_{n+1}'' &= f(u_n) + (u_{n+1} - u_n) \frac{\partial f(u_n)}{\partial u_n}, \\
    u_{n+1}(0) &= u_{n+1}(l) = 0.
\end{align*} \hspace{1cm} (2)

They can be solved by producing a particular solution and two independent solutions of the homogeneous equations numerically and determining the constant multipliers of the homogeneous solutions so as to satisfy the boundary conditions. The sequence of functions \(u_0(x), u_1(x), u_2(x), \ldots\) converges quadratically to the solution \(u(x), 0 \leq x \leq l\), under appropriate conditions on the function \(f(u)\); cf. Ref. 1.
Picard's method for finding the \((n+1)^{\text{st}}\) approximation requires the solution of the linear system
\[
\begin{align*}
  u_{n+1}^\varepsilon &= f(u_n), \\
  u_{n+1}^\varepsilon(0) &= u_{n+1}^\varepsilon(1) = 0.
\end{align*}
\] (3)

It is a first order method, so that if the \((n+1)\)st approximation is
\[
u_{n+1} = u + \delta_n,
\] (4)
where \(\delta_n\) is the discrepancy, then
\[
||\delta_{n+1}|| \sim K ||\delta_n||.
\] (5)

If \(K < 1\), the sequence of approximating functions converges linearly or "geometrically."

\textbf{II. SKETCH OF METHOD}

Let us assume that we have a current approximation \(U_0(x)\), \(0 \leq x \leq 1\). We would like to find a new approximation using a variation of Picard's method. The method which we propose has the advantage of quadratic convergence without requiring any partial differentiation which could require a major calculation in a larger problem. Let us compute the functions \(u_1(x)\) and \(u_2(x)\) by means of the equations
\[
\begin{align*}
  u_1^\varepsilon &= f(u_0), \quad u_1(0) = u_1(1) = 0, \\
  u_2^\varepsilon &= f(u_1), \quad u_2(0) = u_2(1) = 0.
\end{align*}
\] (6)
The application of Eqs. (4) and (5) to \( u_1 \) and \( u_2 \) leads to the following relations:

\[
\| u_1 - u \| \sim K \| U_0 - u \|,
\]

\[
\| u_2 - u \| \sim K \| u_1 - u \|.
\]

Next we write the constant \( K \) in the approximate forms

\[
K \approx \frac{u_1 - U_1}{U_0 - U_1},
\]

and also,

\[
K \approx \frac{u_2 - U_1}{u_1 - U_1},
\]

where the function \( U_1 \) is the prediction of \( u(x) \). Equating these two expressions for \( K \) leads to the extrapolation formulas

\[
U_1 = \frac{U_0 u_2 - u_1^2}{U_0 - 2u_1 + u_2}, \quad (7)
\]

\[
U_1 = u_2 - \frac{(u_1 - u_2)^2}{U_0 - 2u_1 + u_2}. \quad (8)
\]

Equation (7) has the same form as the first order transform of Shanks. (2).

**III. EXAMPLE**

This extrapolation technique was applied to the case

\[
f(u) = e^u.
\]
An initial approximation was computed with $u(0) = 0$, $u'(0) = -1$, and the differential Eq. (1). The integration step length was .01. The results of this experiment are given in the table, together with the correct solution $u(x)$.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$u_0(x)$</th>
<th>$u_1(x)$</th>
<th>$u_2(x)$</th>
<th>$u(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.1</td>
<td>-0.095159</td>
<td>-0.041328</td>
<td>-0.041487</td>
<td>-0.041436</td>
</tr>
<tr>
<td>0.2</td>
<td>-0.181212</td>
<td>-0.073067</td>
<td>-0.073337</td>
<td>-0.073269</td>
</tr>
<tr>
<td>0.3</td>
<td>-0.258912</td>
<td>-0.095566</td>
<td>-0.095961</td>
<td>-0.095800</td>
</tr>
<tr>
<td>0.4</td>
<td>-0.326585</td>
<td>-0.109061</td>
<td>-0.109274</td>
<td>-0.109238</td>
</tr>
<tr>
<td>0.5</td>
<td>-0.391654</td>
<td>-0.113677</td>
<td>-0.113708</td>
<td>-0.113704</td>
</tr>
<tr>
<td>0.6</td>
<td>-0.447657</td>
<td>-0.109433</td>
<td>-0.109209</td>
<td>-0.109238</td>
</tr>
<tr>
<td>0.7</td>
<td>-0.497265</td>
<td>-0.096231</td>
<td>-0.095746</td>
<td>-0.095800</td>
</tr>
<tr>
<td>0.8</td>
<td>-0.540786</td>
<td>-0.073855</td>
<td>-0.073205</td>
<td>-0.073269</td>
</tr>
<tr>
<td>0.9</td>
<td>-0.579481</td>
<td>-0.041950</td>
<td>-0.041387</td>
<td>-0.041436</td>
</tr>
<tr>
<td>1.0</td>
<td>-0.610565</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Note that a single iteration gave a very great improvement over the initial approximation. The time required to do these four iterations on the IBM 7090 computer was about 20 seconds.

The FORTRAN II programs are listed below.

REFERENCES


* LIST  
CPICARD  
PICARD'S METHOD + 2-PT. B.V. PROBLEMS  
C  
COMMON T,NMAX,KMAX,N1,NPRNT,ZERO,U,IFLAG,DELTA,UPREV  
DIMENSION T(100),ZERO(2),U(201)  
DIMENSION P1(201),P2(201)  
DIMENSION U1(201)  
C  
1 READ1000,NMAX,KMAX,N1,NPRNT  
PRINT910  
PRINT900,NMAX,KMAX,N1,NPRNT  
RFAD1001,DELTA,(ZERO(I),I=1,2)  
PRINT901,DELTA,(ZERO(I),I=1,2)  
C  
K=0  
PRINT902,K  
CALL START  
C  
K ITERATIONS, EACH A DOUBLE APPLICATION OF PICARD'S METHOD  
C  
DO 100K=1,KMAX  
PRINT902,K  
IFLAG=1  
C  
APPLICATION 1  
T(2)=0.  
T(3)=DELTA  
T(4)=0.  
T(5)=0.  
UPREV=U(1)  
CALL INT(T,2,N1,0,0,0,0,0,0)  
P1(1)=T(4)  
C  
DO 2 N=2,NMAX  
UPREV=U(N-1)  
CALL INTM  
2  
P1(N)=T(4)  
C  
B1=-P1(NMAX)  
C  
APPLICATION 2  
T(2)=0.  
T(4)=0.  
T(5)=0.  
U1(1)=P1(1)  
UPREV=U1(1)  
CALL INT(T,2,N1,0,0,0,0,0,0)  
P2(1)=T(4)  
C  
DO 3 N=2,NMAX  
UPREV=U1(N-1)  
CALL INTM  
U1(N)=P1(N) + B1*T(2)  
3  
P2(N)=T(4)
NEW APPROXIMATION

PRINT903
U(1)=0.
T(2)=0.
PRINT904,T(2)*U(1)
N=1

5 DO 53 M=1,NPRINTF
N=N+1
T(2)=T(2)+DELTA
U2=P2(N) + B2*T(2)
IF(N-NMAX)51,52,52
51 U(N)=U2 - (U1(N)-U2)**2 / (U(N)-2.*U1(N)+U2)
GO TO 53
52 U(NMAX)=0.
53 CONTINUE

PRINT904,T(2)*U(N)
IF(N-NMAX)5,1,100

100 CONTINUE

GO TO 1

1000 FORMAT(6I12)
1001 FORMAT(6E12.8)
900 FORMAT(6I20)
901 FORMAT(6E20.8)
902 FORMAT(10H1ITERATION, I3)
903 FORMAT(1H018XIHT,26XH(U(T) )
904 FORMAT(10XF10.4, E30.8)
910 FORMAT(37H1PICARD'S METHOD + 2-PT. B.V.PROBLEMS )
END
LIST
SUBROUTINE START
COMMON T*NMAX*KMAX,N1,NPRNT,UZER0*U,IFLAG,DELTA,UPREV
DIMENSION T(1:0),UZER0(2),U(2:1)
C
C ACTUAL EQS.
C
IFLAG=2
T(2)=0.
T(3)=DELTA
T(4)=UZER0(1)
T(5)=UZER0(2)
CALL INT(T,2,N1,0,0,0,0,0,0)
N=1
U(1)=T(4)
PRINT903
PRINT904,T(2)*U(1)
C
DO 6 M=1,NPRNT
CALL INTM
N=N+1
6 U(N)=T(4)
C
PRINT904,T(2)*U(N)
IF(N-NMAX)5,7,7
7 RETURN
C
903 FORMAT(1H018X1HT,26X4HU(T) /)
904 FORMAT(10XF10.4, E30.8)
END
* LIST
  SUBROUTINE DAUX
  COMMON T,NMAX,KMAX,N1,NPRNT,UZERO,U,IFLAG,DELTAP,UPREV
  DIMENSION T(1:0),UZERO(2),U(201)
  GO TO (1,2)*IFLAG
  C
  C PICARD'S Eqs.
  1 T(6)=T(5)
  T(7)=EXPF(UPREV)
  RETURN
  C
  C ACTUAL Eqs.
  2 T(6)=T(5)
  T(7)=EXPF(T(4))
  RETURN
  END
* DATA
  201
  4
  2
  10
  0.005
  0
  -1